CPE
Expressões Relacionais, Expressões Lógicas e Comandos Condicionais

Departamento de Engenharia Elétrica – UnB

Roteiro

1. Expressões relacionais
2. Expressões lógicas
3. Comandos condicionais
4. Decisão simples e decisão múltipla
5. O comando `switch`

Expressão

- Já vimos que constantes, variáveis e endereços de variáveis são expressões.

Exemplo

```
a = 10;
a = b;
endereco = &a;
```

- Vimos também que operações aritméticas também são expressões.

Exemplo

```
a = 2 + 2;
a = 10 / (float) 3;
a = a + 1;
```

Expressões relacionais

São aquelas que realizam uma comparação

$$A \text{ é } \left\{ \begin{array}{c} \text{maior} \\ \text{menor} \\ \text{igual} \\ \text{diferente} \end{array} \right\} \text{ que } B$$

entre duas expressões e resultam no que chamamos de um *valor-verdade* ou valor *Booleano*.

1. `false` (0), se o resultado é falso
2. `true` (1 ou qualquer número diferente de zero), se o resultado é verdadeiro.

Expressões relacionais em C++

- $<$ *expressao* $>$ == $<$ *expressao* $>$: Retorna verdadeiro quando as expressões forem iguais.
  Ex: `a == b`
- $<$ *expressao* $>$!= $<$ *expressao* $>$: Retorna verdadeiro quando as expressões forem diferentes.
  Ex: `a != b`

Expressões relacionais em C++

- $<$ *expressao* $>$ `>` $<$ *expressao* $>$: Retorna verdadeiro quando a expressão da esquerda tiver valor maior que a expressão da direita. Ex: `a > b`
- $<$ *expressao* $>$ `<` $<$ *expressao* $>$: Retorna verdadeiro quando a expressão da esquerda tiver valor menor que a expressão da direita. Ex: `a < b`

Expressões relacionais em C++

- $<$ *expressao* $>$ `>=` $<$ *expressao* $>$: Retorna verdadeiro quando a expressão da esquerda tiver valor maior ou igual que a expressão da direita.
  Ex: `a >= b`
- $<$ *expressao* $>$ `<=` $<$ *expressao* $>$: Retorna verdadeiro quando a expressão da esquerda tiver valor menor ou igual que a expressão da direita.
  Ex: `a <= b`

Expressões lógicas

Expressões lógicas são aquelas que realizam uma operação lógica (`ou`, `e`, `não`, etc...) e retornam verdadeiro ou falso (como as expressões relacionais).

"Se amanhã estiver chovendo E eu estiver a pé, levarei meu guarda-chuva"

"Se tem macarrão OU frango no jantar, eu fico feliz."

"Se essa discussão continuar, a Marcela NÃO conseguirá estudar."

Expressões lógicas em C++

Operador **E** (AND)

- $<$ *expressao* $>$ && $<$ *expressao* $>$: Retorna verdadeiro quando ambas as expressões são verdadeiras. Sua tabela verdade é:

Op_1	Op_2	Ret
V	V	V
V	F	F
F	V	F
F	F	F

Exemplo

```
a == 0 && b == 0
```

Expressões lógicas em C++

Operador **OU** (OR)

- $<$ *expressao* $>$ || $<$ *expressao* $>$: Retorna verdadeiro quando pelo menos uma das expressões é verdadeiras. Sua tabela verdade é:

Op_1	Op_2	Ret
V	V	V
V	F	V
F	V	V
F	F	F

Exemplo

```
a == 0 || b == 0
```

Expressões lógicas em C++

Operador de **Negação** (NOT)

- ! $<$ *expressao* $>$: Retorna verdadeiro quando a expressão é falsa e vice-versa. Sua tabela verdade é:

Op_1	Ret
V	F
F	V

Exemplo

```
!(a == 0)
```

Simplificações úteis

- `!(a == b)` é equivalente a `a != b`
- `!(a != b)` é equivalente a `a == b`
- `!(a > b)` é equivalente a `a <= b`
- `!(a < b)` é equivalente a `a >= b`
- `!(a >= b)` é equivalente a `a < b`
- `!(a <= b)` é equivalente a `a > b`

Leis de De Morgan

- `!a && !b` é equivalente a `!(a || b)`

a	b	a||b	!(a||b)	!a	!b	!a&&!b
V	V	V	F	F	F	F
V	F	V	F	F	V	F
F	V	V	F	V	F	F
F	F	F	V	V	V	V

- `!a || !b` é equivalente a `!(a && b)`

a	b	a&&b	!(a&&b)	!a	!b	!a||!b
V	V	V	F	F	F	F
V	F	F	V	F	V	V
F	V	F	V	V	F	V
F	F	F	V	V	V	V

O tipo bool

A linguagem C++ possui o tipo `bool` para armazenar valores-verdade `true` ou `false`.

Exemplo

```
bool sentenca1, sentenca2;
sentenca1 = (a>2);
sentenca2 = false;
```

Comandos condicionais

Um comando condicional é aquele que permite decidir se um determinado bloco de comandos deve ou não ser executado, a partir do resultado de uma expressão relacional ou lógica.

Comandos condicionais

- O principal comando condicional da linguagem C++ é o `if`, cuja sintaxe é:

```
if (expressão lógica)
    comando;
```

ou

```
if (expressão lógica) {
    comando 1;
    comando 2;
    ...
    comando n;
}
```

- Os comandos são executados somente se a expressão lógica for verdadeira.

Comandos condicionais

Exemplo

Construa um algoritmo que, dado um valor, determina se ele é ímpar.

Comandos condicionais

O programa abaixo determina se um valor é ímpar.

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main(){
6      int a;
7
8      cin >> a;
9
10     if((a%2) != 0)
11         cout << "O valor e impar.\n";
12
13     return 0;
14
15 }
```

Comandos condicionais compostos

- Uma variação do comando `if` é o `if/else`, cuja sintaxe é:

```
if (expressão lógica) {
    comandos executados se a expressão é verdadeira
} else {
    comandos executados se a expressão é falsa
}
```

Comandos condicionais compostos

Exemplo

Construa um algoritmo que, dado um valor, determina se ele é ímpar ou se é par.

Comandos condicionais compostos

O programa abaixo determina se um valor é ímpar ou par.

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main(){
6      int a;
7
8      cin >> a;
9
10     if((a%2) == 0)
11         cout << "O valor e par.\n";
12     else
13         cout << "O valor e impar.\n";
14
15     return 0;
16
17 }
```

Comandos condicionais compostos

```
if (cond1)
  if (cond2)
    comando1;
else
  comando2;
```

Quando o `comando2` é executado?

Comandos condicionais

```
if (cond1)
  if (cond2)
    comando1;
  else
    comando2;
```

Quando o comando2 é executado?

Comandos condicionais

```
if (cond1) {
  if (cond2)
    comando1;
} else
  comando2;
```

Quando o comando2 é executado?

Decisão simples e decisão múltipla

- Dependendo do problema proposto, o programa pode ser formado por um conjunto muito grande de comandos `if` e expressões lógicas.

Exemplo

Faça um programa que, dada a matrícula, emite uma mensagem se o aluno estiver matriculado na disciplina de CPE.

Decisão simples

Para apenas um aluno, a solução seria:

```
int main () {
    int a;
    std::cin >> a;
    if (a == 10129) {
        std::cout << "O aluno " << a << " esta matriculado\n";
    }
    return 0;
}
```

Decisão múltipla

Para dois alunos, a solução seria:

```
int main () {
    int a;
    std::cin >> a;
    if (a == 10129 || a == 16267) {
        std::cout << "O aluno " << a << " esta matriculado\n";
    }
    return 0;
}
```

Decisão múltipla

- Problema: CPE possui 35 alunos neste semestre.

```
if (a == 2582 || a == 10129 ||
    a == 16267 || ...
    a == 962185) {
    cout << "O aluno " << a << " esta matriculado\n";
}
```

- Teríamos muitas condições a serem testadas.

Decisão múltipla

- Temos um conjunto muito grande de alunos.
- Além disso, fica improdutivo utilizar os operadores lógicos e relacionais que utilizamos anteriormente.
- Podemos tentar diminuir o número de testes realizados?
- Uma construção bem comum é o uso da sequência `if else if`:

```
if (<condição1>)
   <comando>
else if (<condição2>)
   <comando>
...
else if (<condiçãoN>)
   <comando>
```

O comando switch

O objetivo do comando switch é simplificar uma expressão onde uma variável inteira ou caractere deve fazer diferentes operações dependendo exclusivamente de seu valor.

```
switch (variável inteira ou char) {
    case <valor1>:
    comando 1;
    ...
    comando n;
    break;
    case <valor2>:
    comando 1;
    ...
    comando n;
    break;
}
```

O comando switch

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5    unsigned int a;
6    cout << "Matricula: ";
7    cin >> a;  //leitura
8    switch(a) {
9      case 10129:
10       cout << "Maria Candida Moreira Telles\n";
11       break;
12     case 33860:
13       cout << "Larissa Garcia Alfonsi\n";
14       break;
15     case 33967:
16       cout << "Leonardo Kozlowiski Kenupp\n";
17       break;
18   }
19   return 0;
20 }
```

O comando `switch`

- Os comandos começam a ser executados a partir do ponto onde o valor da variável corresponde ao valor antes dos dois pontos (:).
- Executa todos os comandos até que encontre um comando `break` ou que chegue ao final do bloco de comandos do `switch`

Valor padrão

- Você pode utilizar, ao invés de um valor, o valor `default`. A execução dos comandos inicia no comando `default` se nenhum outro valor for correspondente ao valor da variável.

Sintaxe

```
switch (variável inteira) {
    case <valor>:  comandos break;
    default:  comandos
}
```

Valor padrão

```
1  #include <iostream>
2  using namespace std;
3
4  int main () {
5    int a;
6    cin >> a;
7    switch(a) {
8      case 10129:
9          cout << "Maria Candida Moreira Telles\n";
10         break;
11     case 33860:
12         cout << "Larissa Garcia Alfonsi\n";
13         break;
14     default:
15         cout << "O aluno nao esta matriculado\n";
16   }
17   return 0;
18 }
```

Dilbert


YOU SHOULD HIRE ME AS YOUR MANAGEMENT CONSULTANT.
WE'RE A MANAGEMENT CONSULTING FIRM. WE DON'T NEED A MANAGEMENT CONSULTANT TO CONSULT US.
ARE YOU SAYING THAT MANAGEMENT CONSULTING IS WORTHLESS?
NO, I'M SAYING WE ALREADY KNOW EVERYTHING ABOUT MANAGEMENT CONSULTING.
HOW CAN YOU BE SO SURE I CAN'T HELP YOU WHEN YOU DON'T KNOW WHAT MY ADVICE WILL BE?
OKAY, YOU'RE HIRED. WHAT'S YOUR ADVICE?
BEATS ME. I'LL HAVE TO ASK MY MANAGEMENT CONSULTANT.
E-mail: SCOTTADAMS@AOL.COM
© 2007 Scott Adams, Inc./Dist. by UFS, Inc.
www.dilbert.com